Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №8

Технології розробки програмного забезпечення

ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»

Виконала:
студент групи ІА-24
Красношапка Р. О.
Перевірив:
Мягкий М. Ю.

Київ 2024

Зміст

Тема: ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»

Короткі теоретичні відомості

Шаблони проектування "Composite", "Flyweight", "Interpreter" і "Visitor" є структурними або поведінковими патернами, які допомагають вирішувати різні типи проблем у розробці програмного забезпечення.

Composite (Композиція) є структурним шаблоном проектування, який дозволяє об'єднати об'єкти в дерево так, що клієнти можуть працювати з окремими об'єктами та їх композиціями однаково. Це дозволяє будувати складні структури, де компоненти можуть бути як простими, так і складеними, і забезпечує зручний інтерфейс для їх взаємодії.

Flyweight (Летючий вагон) є структурним шаблоном проектування, що дозволяє економити ресурси за рахунок спільного використання великої кількості об'єктів, що мають однакові дані або поведінку. Flyweight зберігає спільні властивості об'єктів, а специфічні дані передає через контекст, дозволяючи значно зменшити використання пам'яті.

Interpreter (Інтерпретатор) є поведінковим шаблоном проектування, який визначає граматику певної мови та надає спосіб інтерпретації виразів цієї мови. Цей шаблон зазвичай використовується для створення інтерпретаторів мов або для реалізації специфічних бізнес-правил, які можуть бути представлені у вигляді виразів чи синтаксичних дерев.

Visitor (Візитер) є поведінковим шаблоном проектування, що дозволяє додавати нову поведінку до об'єктів без зміни їх класів. Візитер використовується для обробки об'єктів різних класів за допомогою однієї операції, що дає можливість додавати нові операції до структури об'єктів, не змінюючи їх. Це підвищує гнучкість і дозволяє легко додавати нові функціональності до існуючих класів.

Хід роботи

Project Management software (proxy, chain of responsibility, abstract factory, bridge, flyweight, client-server) Програмне забезпечення для управління проектами повинно мати наступні функції: супровід завдань/вимог/проектів, списків команд, поточних завдань, планування за методологіями agile/kanban/rup (включаючи дошку завдань, ітерації тощо), мати можливість прикріплювати вкладені файли до завдань та посилатися на конкретні версії програми, зберігати виконувані файли для кожної версії.

Основні принципи та застосування Flyweight (Летючий вагон):

Шаблон проектування **Flyweight (Летючий вагон)** є структурним патерном, що допомагає значно зменшити використання пам'яті та підвищити ефективність програм, які працюють з великою кількістю об'єктів, що мають спільні характеристики. Цей шаблон дозволяє спільно використовувати однакові частини об'єктів, зберігаючи лише унікальні, змінювані дані для кожного екземпляра. Це дозволяє значно знизити дублювання інформації та заощадити пам'ять, що особливо важливо при роботі з великою кількістю схожих елементів.

Принцип роботи Flyweight полягає в розділенні стану об'єкта на дві частини: **внутрішній (intrinsic)** стан, який спільно використовують всі об'єкти, та **зовнішній (extrinsic)** стан, що є специфічним для кожного об'єкта і передається під час виконання операцій. Це дозволяє мінімізувати витрати на створення об'єктів, зберігаючи спільні частини в одному місці для повторного використання.

Шаблон Flyweight застосовується в ситуаціях, де необхідно працювати з великою кількістю схожих об'єктів. Наприклад, в графічних системах, де одні й ті самі пікселі або елементи інтерфейсу можуть бути спільно використані. Також цей патерн корисний у відеоіграх, веб-додатках або базах даних, де зберігається велика кількість схожих елементів з лише незначними відмінностями.

Завдяки використанню Flyweight можна значно зменшити витрати пам'яті, підвищити ефективність роботи системи та забезпечити масштабованість, навіть коли обсяг даних дуже великий.

Реалізація шаблону проєктування для майбутньої системи

Шаблон **Flyweight (Летючий вагон)** є структурним шаблоном проєктування, який дозволяє значно знизити споживання пам'яті шляхом спільного використання об'єктів, що мають однакові характеристики. Це досягається за рахунок розділення об'єкта на внутрішні та зовнішні стани. Внутрішній стан є спільним для всіх об'єктів, а зовнішній — змінюється залежно від конкретного контексту.

Шаблон **Flyweight** особливо корисний у системах, де створюється велика кількість об'єктів, які мають схожі характеристики, але відрізняються деякими параметрами. Замість того, щоб створювати новий об'єкт для кожного випадку, можна створювати лише один спільний об'єкт і зберігати змінний стан окремо.

У майбутній системі застосування шаблону **Flyweight** дозволить знизити витрати пам'яті при створенні великої кількості подібних об'єктів. Наприклад, у випадку обробки великої кількості однакових компонентів або елементів, які мають спільні властивості (наприклад, дані користувачів, географічні координати тощо), шаблон дозволяє використовувати спільний внутрішній стан для багатьох об'єктів, зберігаючи лише унікальні значення для кожного з них у зовнішньому стані. Це значно знижує кількість створених об'єктів і зменшує навантаження на систему.

```java
package org.example.projectmanagement.flyweight;

import org.example.projectmanagement.models.Version;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;

2 usages  QUIRINO *
public class VersionFlyweightFactory {
    3 usages
    private static final Map<String, Version> flyweights = new HashMap<>();

    1 usage  new *
    public static Version getVersion(String versionName, String releaseDate, String executableFilePath) {
        String key = generateKey(versionName, releaseDate, executableFilePath);
        if (!flyweights.containsKey(key)) {
            flyweights.put(key, Version.builder()
                    .versionName(versionName)
                    .releaseDate(LocalDate.parse(releaseDate))
                    .executableFilePath(executableFilePath)
                    .build());
        }
        return flyweights.get(key);
    }

    1 usage  new *
    private static String generateKey(String versionName, String releaseDate, String executableFilePath) {
        return String.join( delimiter: ":", versionName, releaseDate, executableFilePath);
    }
}
```

Рис. 1 – Код інтерфейсу VersionFlyweightFactory

```java
package org.example.projectmanagement.models;

import ...

14 usages  QUIRINO
@Data
@Builder
@Document(collection = "versions")
@AllArgsConstructor
@NoArgsConstructor
public class Version {
    @Id
    private String versionId;
    private String versionName;
    private LocalDate releaseDate;
    private String executableFilePath;
}
```

Рис. 2 – Код класу Version

Рис. 3 – Приклад використання VersionFlyweightFactory

Принцип роботи патерну Flyweight у проекті полягає в оптимізації використання пам'яті шляхом повторного використання об'єктів, які мають однакові властивості. Це досягається за допомогою фабрики Flyweight, яка зберігає створені об'єкти та повертає існуючі, якщо вони вже були створені з такими ж параметрами. Ось як це працює у вашому проекті:

Зберігання Flyweight об'єктів: Використовується HashMap для зберігання об'єктів Version.

Генерація ключа: Для кожної комбінації versionName, releaseDate та executableFilePath генерується унікальний ключ.

Отримання/створення об'єкта: Метод getVersion перевіряє, чи існує об'єкт Version з даним ключем. Якщо ні, створюється новий об'єкт за допомогою патерну Builder і зберігається в мапі.

Зображення структури шаблону

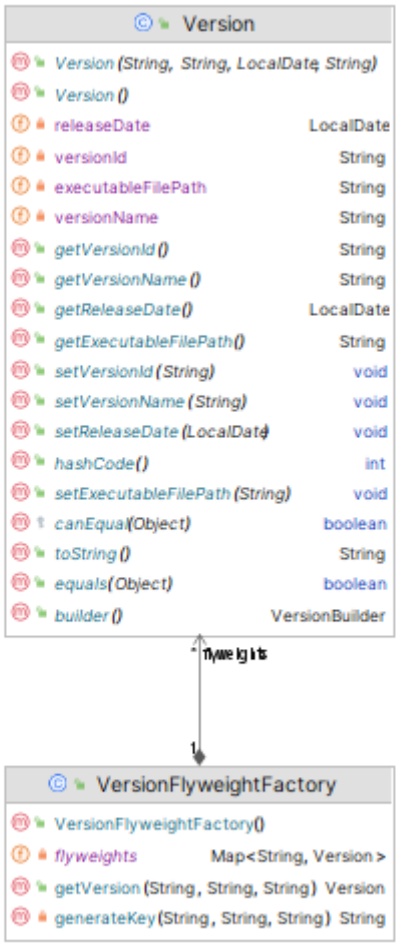

Рис. 4 – Структура шаблону

Посилання на репозиторій:

https://github.com/QUIRINO228/projectManagmentSoftware

Висновок: У результаті реалізації шаблону Flyweight було продемонстровано ефективність цього патерна для оптимізації використання пам’яті та підвищення продуктивності системи. Шаблон дозволяє розділяти стан об’єктів на спільний (внутрішній) та змінний (зовнішній), що забезпечує можливість повторного використання об’єктів із загальними характеристиками.

Завдяки цьому підходу система знижує кількість створюваних об'єктів і оптимізує використання ресурсів, оскільки однакові об'єкти зберігаються в єдиному екземплярі. Це особливо ефективно у випадках, коли система працює з великою кількістю схожих елементів, таких як графічні об’єкти, текстові символи чи дані, що повторюються.

Шаблон Flyweight забезпечує масштабованість і простоту додавання нових функціональностей, оскільки спільний стан об’єктів централізовано керується, а змінний стан зберігається окремо. Завдяки цьому система легко адаптується до нових вимог, залишаючись чистою, ефективною і легкою для підтримки.